WORKING PAPER NO. 03-17/R
AN EMPIRICAL LOOK AT SOFTWARE PATENTS

James Bessen*
Research on Innovation and
Boston University School of Law (Visiting Researcher)

Robert M. Hunt**
Federal Reserve Bank of Philadelphia

First Draft: August 2003
This Draft: March 2004

* jbessen@researchoninnovation.org

** Ten Independence Mall, Philadelphia, PA 19106. Phone: (215) 574-3806. Email: bob.hunt@phil.frb.org

Thanks to Peter Bessen of May8Software for providing a software agent to acquire our patent database and Annette Fratantaro for her work with the Compustat data set. Also thanks to John Allison, Tony Breitzman and CHI Research, Iain Cockburn, Mary Daly, Dan Elfenbein, Terry Fisher, Bronwyn Hall, Joachim Henkel, Brian Kahin, David Mowery, Leonard Nakamura, Cecil Quillen, Eric von Hippel, Rosemarie Ziedonis and seminar participants at APPAM, Berkeley, EPIP Munich, Federal Reserve Banks of Philadelphia and San Francisco, the Federal Reserve System Applied Micro meetings, Harvard, IDEI, MIT, NBER, and OECD.

AN EMPIRICAL LOOK AT SOFTWARE PATENTS

James Bessen

Research on Innovation and
Boston University School of Law (Visiting Researcher)

Robert M. Hunt

Federal Reserve Bank of Philadelphia

March 2004

Abstract:

U.S. legal changes have made it easier to obtain patents on inventions that use software. Software patents have grown rapidly and now comprise 15 percent of all patents. They are acquired primarily by large manufacturing firms in industries known for strategic patenting; only 5 percent belong to software publishers. The very large increase in software patent propensity over time is not adequately explained by changes in R&D investments, employment of computer programmers, or productivity growth. The residual increase in patent propensity is consistent with a sizeable rise in the cost effectiveness of software patents during the 1990s. We find evidence that software patents *substitute* for R&D at the firm level; they are associated with *lower* R&D intensity. This result occurs primarily in industries known for strategic patenting and is difficult to reconcile with the traditional incentive theory of patents.

Keywords: Software, Patents, Innovation, Technological Change

JEL classification: O34, D23, L86

Introduction

Federal courts, and to a lesser extent the U.S. Patent and Trademark Office (USPTO), dramatically changed standards for patenting software-related inventions over the last three decades. During the 1970s, federal court decisions typically described computer programs as mathematical algorithms, which are unpatentable subject matter under U.S. law.[1] Systems using software could be patented, but only if the novel aspects of the invention did not reside entirely in the software.[2] At this time, the U.S. Congress considered the question of patenting software and instead opted to protect computer programs under copyright law.[3]

But after the Supreme Court decision in *Diamond v. Diehr* in 1981,[4] a series of court and administrative decisions gradually relaxed the subject matter exception that restricted the patenting of software-related inventions. The 1994 decision *In re Alapat* eliminated much of the remaining uncertainty over the patentability of computer programs.[5] During this same period, new legislation and other court decisions lowered standards for obtaining patents in general, while strengthening aspects of patent enforcement.

This paper explores the general characteristics of software patenting over the last two decades, paying particular attention to the rapid growth in software patenting and the effect of this growth on R&D. We construct our own definition of a *software patent* (there is no official definition) and assemble a comprehensive database of all such patents. In Section I we describe this process, and the process of matching these patents to firm data in the Compustat database. In Section II we summarize the general characteristics of this data. We find that over 20,000 software patents are now granted each year, comprising about 15 percent of all patents. Compared with other patents,

---

[1] See, for example, the Supreme Court decision in *Gottschalk v. Benson*, 409 U.S. 63 (1972).

[2] *Parker v. Flook* 437 U.S. 584 (1978).

[3] U.S. Copyright law was amended in 1976, and more explicitly in 1980, to include computer programs. See H. Rpt. No. 94-1476 (1976) and P.L. 96-517 (94 Stat 3028). There is a voluminous literature on the merits of different forms of intellectual property protection for computer programs. See, for example, Dam (1995), Graham and Zerbe (1996), and Samuelson et al. (1994).

[4] 450 U.S. 175 (1981).

[5] 33 F.3d 1526 (Fed. Cir. 1994).

software patents are more likely to be assigned to firms, especially larger U.S. firms, than to individuals. They are also more likely to have U.S. inventors. Surprisingly, most software patents are assigned to manufacturing firms and relatively few are actually assigned to firms in the software publishing industry (SIC 7372). Most software patents are acquired by firms in industries that are known to accumulate large patent portfolios and to pursue patents for strategic reasons (computers, electrical equipment, and instruments). These large inter-industry differences remain even after we control for R&D, software development effort, and other factors.

In Section III we perform regressions that explore the "propensity to patent" software inventions. This builds on the model of Hall and Ziedonis (2001) which, in turn, builds on the empirical literature of "patent production functions" (including Scherer 1965, Bound et al. 1984, Pakes and Griliches 1984, and Griliches, Hall, and Hausman 1986). We find a dramatic growth in software patent propensity even after controlling for R&D, employment of computer programmers, and other factors. This growth is quite similar to the remarkable growth in patent propensity that Hall and Ziedonis (2001) found in the semiconductor industry. Productivity-based explanations are unlikely to account for even half of the rise in software patent propensity. The pattern of the residual increase is consistent with the explanation that changes in patent law made software patents significantly more cost effective. We also find that industries known for strategic patenting have much higher patent propensities.

Section IV explores the effect of software patenting on R&D. According to the traditional incentive theory, making software patents more cost effective should increase the profitability of firms that make software inventions. This, in turn, should induce them to increase their R&D spending. To test whether this in fact happened, we use a well-established empirical technique for estimating elasticities of factor substitution. We show that the incentive hypothesis can be re-stated as the hypothesis that R&D and patents are complements. This means that increases in the appropriability of software should lead to greater R&D intensity. We find that this is not the case, however. Firms that increased their software patenting relative to their overall level of patenting tended to *decrease* their R&D intensity relative to other firms. This result is robust to a variety of

econometric and other considerations. But, again, this effect is concentrated in the industries known for strategic patenting.

In Section V, we note that while our empirical analysis does not identify the specific causal mechanisms at work, these results are difficult to reconcile with the traditional incentive hypothesis. Strategic patenting provides an explanation for a rise in patent propensity together with an apparent substitution away from R&D: firms may be engaged in a patent "arms race." The prominence of certain industries—known for strategic patent behavior in other contexts—in our empirical results may not be coincidental. Section VI concludes.

I. Background and Data

A. Changing Legal Treatment and Strategic Patenting Industries

The erosion of the subject matter exception for computer programs occurred against a backdrop of broader changes in patent law following the creation of a unified appeals court for patents suits in 1982 (see Hall and Ziedonis 2001, for a nice summary). The court raised the evidentiary standards required to challenge patent validity and tended to broaden the interpretation of patent scope (Rai 2003, Merges 1997). The court relaxed the standards for evaluating whether or not an invention is obvious to practitioners skilled in the art (Cooley 1994, Dunner et al. 1995, Hunt 1999, Lunney 2001). The court was also more willing to grant preliminary injunctions to patentees (Cunningham 1995, Lanjouw and Lerner 2001) and to sustain large damage awards (Merges 1997, Kortum and Lerner 1999). Finally, plaintiff success rates in patent infringement suits have increased substantially (Lerner 1995).

Some of these changes made patents "stronger," in the sense that patents became more likely to be upheld in court or more effectively enforced. Others made patents "cheaper," in the sense that lower patentability standards reduced the effort required to obtain a patent on a given invention. Together, these changes made patents more *cost effective* than before, generating more appropriability per dollar invested in obtaining and asserting them.

If patents did become more cost effective, the change is likely greater for software than for other inventions, for two reasons. First, the presumption that computer programs could not be patented was largely reversed by the mid-1990s. Second, a number of other legal decisions relaxed the "enablement" requirement for software patents. Under U.S. patent law, filers are required to provide detailed instructions explaining how the invention works. It is supposed to be a "best mode" example of all of the patent claims. For software patents and business methods, it seems the courts have largely eliminated this requirement (Burk 2002, Burk and Lemley 2002).[6] In the words of an IBM patent attorney, "[the patent standard] currently being applied in the U.S. invites the patenting of ideas that may have been visualized as desirable but have no foundation in terms of the research or development that may be required to enable their implementation" (Flynn, 2001). The combined effect of these regulatory changes is that software patents appear to have gained greater appropriability and became less costly to obtain in absolute terms over time and also possibly relative to other patents.

Yet the software industry was highly innovative and growing rapidly well before software patents became commonplace. Nominal investment in software grew 16 percent per annum during the 80s (and 11 percent per annum during the 90s, Grimm and Parker 2000). This innovativeness is important for two reasons. First, in interpreting results below, the growing use of software is an important factor. Second, given this history, it is not at all clear that patent protection was essential for innovation in this industry.

This is not unusual. When surveyed, American firms in a number of other innovative industries (including semiconductors and precision instruments) rate patents as a *relatively* less effective form of appropriability (Levin et al. 1987, Cohen, Nelson, and Walsh 2000). Instead, they cite lead time advantages, learning curves, complementary sales and service and secrecy as generally more important sources of appropriability.

---

[6] Reviewing case law, Burk and Lemley (2002, p. 1162) write: "For software patents, however, a series of recent Federal Circuit decisions has all but eliminated the enablement and best mode requirements. In recent years, the Federal Circuit has held that software patents need not disclose source or object code, flow charts, or detailed descriptions of the patented program. Rather, the court has found high-level functional description sufficient to satisfy both the enablement and best mode doctrines." See also Cohen and Lemley (2001) on the different treatment of software patents.

Yet even in industries where patents are rated as ineffective, we sometimes observe that firms sometimes acquire large patent portfolios. These industries, including the computer, electrical equipment and instruments industries, are also found to account for a major share of the growth in patenting in recent years (Hall 2003). Some researchers have suggested that firms in these industries may patent heavily in order to obtain strategic advantages, including advantages in negotiations, cross-licensing, blocking competitors, and preventing suits (Levin et al. 1987, fn 29, and Cohen, Nelson, and Walsh 2000). In principle, strategic patenting can arise whenever individual products involve many patentable inventions and the cost of obtaining patents is sufficiently low (see Bessen 2003 for a theoretical model). Firms may acquire large numbers of patents so that even if they have an unsuccessful product, they can hold up rivals, threatening litigation. Innovative firms may acquire "defensive" patent portfolios to make a credible counter-threat. The outcome may involve the cross-licensing of whole portfolios, where firms agree not to sue each other and those firms with weaker portfolios pay royalties (Grindley and Teece 1997).

So, in what follows, it is natural for us to be alert to the possibility that software patents may also be acquired for strategic purposes and we will find distinctive behavior in the industries known for strategic patenting. An important implication of strategic patenting is that policy changes that "strengthen" patents (or make them cheaper to acquire) can lead to a kind of "Prisoner's Dilemma" game that actually *decreases* the private incentive to engage in R&D.

B. What Is a Software Patent?

How many software patents are being granted? Although the patent office maintains a system for classifying patents, this system does not distinguish whether the underlying technology is software or something else. Researchers must construct their own definitions.

Some observers have sought to identify "pure" software patents where the invention is completely embodied in software (e.g. Allison and Lemley 2000). We prefer not to use such a definition for two reasons. First, beginning with *Diamond v. Diehr*,

attorneys have drafted software patents in such a way that they did not necessarily *appear* to be patents on software.[7] This makes the determination of a "pure" software patent somewhat arbitrary and impractical for a comprehensive database. Second, we do not necessarily assume that the subject matter exclusion is the *only* difference between software patents and other patents. For example, we noted above that software patents are subject to a different "enablement" requirement. So it is useful to study a somewhat broader range of patents in any case.

Our concept of software patent involves a logic algorithm for processing data that is implemented via stored instructions; that is, the logic is not "hard-wired." These instructions could reside on a disk or other storage medium or they could be stored in "firmware," that is, a read-only memory, as is typical of embedded software. But we want to exclude inventions that involve only off-the-shelf software—that is, the software must be at least novel in the sense of needing to be custom-coded, if not actually meeting the patent office standard for novelty.

1. Identifying software patents

How can we identify patents that fit this description? Griliches (1990) reviews the two main techniques that researchers have used to assign patents to an industry or technology field: 1) using the patent classification system developed by the patent office; and 2) reading and classifying individual patents. In this paper, we use a modification of the second technique.

We began by reading a random sample of patents, classifying them according to our definition of software, and identified some common features of these patents. We used these to construct a search algorithm to identify patents that met our criteria. We used this algorithm to perform a keyword search of the U.S. Patent Office database, which identified 130,650 software patents granted in the years 1976 to 1999. Next, to

[7] For instance, Cohen and Lemley (2001, p. 9) argue "The Diehr decision and its appellate progeny created what might be termed 'the doctrine of the magic words.' Under this approach, software was patentable subject matter, but only if the applicant recited the magic words and pretended that she was patenting something else entirely." In *Diamond v. Diehr*, the Supreme Court ruled that an invention using temperature sensors and a computer program to calculate the correct curing time in an otherwise

validate the accuracy of this algorithm, we compared the results of our search against a random sample of 400 patents which we had read and classified into software patents and other patents. We also compared our results to samples and statistics generated by other researchers. The details of the algorithm and characteristics of the sample are described in the Appendix.

Compared to our random sample of 400 patents, this algorithm had a false positive rate of 16 percent (that is, 16 percent of the patents the algorithm said were software patents, were not) and a false negative rate of 22 percent (that is, it failed to identify 22 percent of the patents we categorized as software patents).

We performed a number of other checks. First, we compared our list of software patents to a set of 330 software and Internet patents identified in research conducted for the papers by Allison and Lemley (2000) and Allison and Tiller (2003).[8] These patents were identified by reading a larger number of patents, but applying a more narrow definition of software inventions (again, where the invention is completely embodied in software).[9] Virtually all (92 percent) of the software patents identified by Allison were categorized as software patents by our algorithm. Thus our false negative rate for "pure" software patents appears to be quite small (8 percent). Second, using statistics generated by the research in Allison and Lemley (2000), we calculated an upper bound on the comparable false positive rate of 26 percent.[10] Given that we are using a broader definition of software patent, it is reassuring to find that this number is not much larger

---

conventional process of molding rubber goods could be patented. We treat this as a software patent; Allison and Lemley would not.

[8] Thanks to John Allison for sharing his data with us. The data used in Allison and Lemley (2000) are based on reading 1,000 randomly selected patents issued between mid 1996 and mid 1998. That data was augmented in Allison and Tiller (2003) by examining 2,800 patents issued between 1990 and 1999 identified via a keyword search (for the terms *Internet* or *world wide web*) restricted to patents included in classes 705, 707, or 709. Note that in the Allison and Tiller taxonomy, internet business method patents are a subset of software patents.

[9] Both papers state the following: "Another researcher might include within the Software classification those inventions in which the algorithms are embodied in chips, but we have chosen to include within our definition of Software only those inventions that consist purely of software that is not embodied in hardware."

[10] Allison reports identifying 92 pure software patents in the sample of 1,000 evaluated in that paper (private communication). This is somewhat higher than the 76 reported in the published version of the paper. For the same years, the ratio of software patents to total patents calculated using our algorithm was 11.5%. Thus an upper bound on the false positive rate is (.115-.092*.92)/.115 = 26%.

than the false positive rate generated when using our own random sample. Although the algorithm does make errors, it performs reasonably well, and it seems unlikely that it introduces significant biases to our patent counts or regression coefficients.

2. Using Patent Classes to Identify Software Inventions

Why did we use this rather laborious method rather than simply counting patents in certain patent classifications? First, in a longitudinal study patent classes are problematic because the classification system changes over time and the patent office continually re-classifies issued patents. Moreover, lawyers are known to draft patents so that they avoid falling into certain classes in order to influence the examiner's prior art search or some other aspects of the examination (Lerner 2004, p. 19).

Second, economists have long recognized the poor correspondence between patent classes and economic concepts of industry or technology (see, for example, Schmookler 1966, Scherer 1982a, Scherer 1984, Soete 1983, and Griliches 1990). Patent classes are not designed with social scientists in mind but are used primarily to aid prior art search. Although patent classes can be used effectively when they are confined to select sets of well-defined subclasses (e.g., Schmookler 1966, Lerner 2004), or when classifications are statistically distributed over industries (e.g., Silverman 1999), or when taken as loosely representative (e.g., Graham and Mowery 2003), a definition based on patent classes is likely to introduce significantly more inaccuracies than the approach chosen in this paper.[11]

In the case of the U.S. classification system, there are no patent classes for software *per se*. Instead, software inventions are included in functional categories along with hardware inventions. For instance, one class includes "arrangements for producing a permanent visual representation of output data." This is a functional description that

[11] Allison and Lemley (2000) and Allison and Tiller (2003) also reject the idea of using patent classifications to identify software patents. According to Griliches (1990), a patent class is "based primarily on technological and functional principles and is only rarely related to economists' notions of products or well-defined industries (which may be a mirage anyway). A subclass dealing with the dispensing of liquids contains both a patent for a water pistol and for a holy water dispenser. Another subclass relating to the dispensing of solids contains patents on both manure spreaders and toothpaste tubes" (Griliches 1990, p. 1666).

includes software programs, hardware computer displays, and even electric and mechanical signs that pre-date electronic computers.

Still, we did examine the efficacy of using the patent classification approach for identifying software inventions, as proposed in Graham and Mowery (2003). In that paper, the authors identified a number of subdivisions of the International Patent Classification system (IPC) where many patents assigned to large U.S. software companies may be found.[12] We compared the list of software patents identified by this approach with our random sample of 400 patents. The results were significantly worse than our own algorithm: a 30 percent false positive rate and a 74 percent false negative rate. We also found that this definition would exclude half of the patents obtained by the top 200 publicly-traded software firms during the 1990s, and a majority of the pure software and Internet patents identified in Allison and Tiller (2003). In contrast, our definition accounts for about 4/5 of all patents obtained by the top 200 software firms in the 1990s.

Nevertheless, to check whether our main results were robust to our choice of algorithm, we ran most of our tabular analyses and regressions below using the Graham-Mowery definition of software patent. Although standard errors were predictably higher, the results were broadly similar. For example, the distribution across industries was similar and patents and R&D were found to be statistically significant substitutes. Although Graham and Mowery's approach is useful for obtaining a rough impression, it is not the best technique for a more comprehensive study such as ours, and attempts to improve its coverage by adding more classes are likely to increase the false positive rate.

To summarize, our algorithm has a positive error rate. For this reason, one can find patents that our algorithm incorrectly classifies as software patents (see Hahn and Wallsten 2003). But the rate of false positives is reasonably low and the algorithm appears to be substantially more accurate than at least one alternative based on patent classes. Moreover, the evidence suggests that there is no *systematic* bias in these errors: our main results hold even when we use a very different definition of software patent.

[12] The subdivisions include G06F 3/, 5/, 7/, 9/, 11/, 13/, and 15/; G06K 9/, and 15/; and H04L 9/.

C. The Matched Sample

We also explore the characteristics of the firms that obtained software patents and, in particular, the relationship between software patenting behavior and firm R&D performance. To do this, we matched a large portion of both software patents and other patents to firms in the 1999 vintage of the Compustat database.

Our main population of interest consists of U.S.-owned public firms that perform R&D. This group performs a large share of domestic R&D and it should provide a relatively stable group for comparison over time. But it does limit the relevance of our conclusions to this group, however, and so our analysis has little to say about start-up firms, individuals, universities, etc.

We begin by matching our patents to firms (i.e. the assignees) using the NBER Patent Citations Data File (Hall, Jaffe and Trajtenberg 2001a).[13] This data set matches patents to the 1989 vintage of firms contained in Compustat, so we do a variety of things to supplement those matches:

1. We added the largest 25 publicly traded software firms ranked by sales (only one of which is included in the NBER file).
2. We merged the data set with a set of firm-patent matches provided to us by CHI Research.[14] That data encompasses most of the significant patenting firms (public or private) over the last 25 years.
3. Using data contained in Compustat, we identified 100 of the largest R&D performers in 1999 that were not already included in our data set. We matched these firms and their subsidiaries to their patents using a keyword search on the USPTO web site.

[13] To be precise, we match patent numbers in our data set with those found in the NBER data set. Where available, we use the firm CUSIP assigned by NBER to obtain financial data from Compustat.

[14] We again match by patent number and use the firm CUSIP assigned by CHI. Details on CHI's proprietary data is described at http://www.chiresearch.com/information/customdata/patdata.php3. We are grateful to Tony Breitzman for sharing this data with us.

The final file held 4,792 distinct subsidiaries and 2,043 parent firms from 1980 to 1999, an improvement of 1,230 subsidiaries and 305 firms over the NBER database for the same period.[15]

To test the coverage of this matched sample, we compared it to the target population in Compustat (that is, all U.S. firms that publicly report financials with positive R&D). The matched sample performed 91 percent of the deflated R&D in the Compustat file over this period and accounted for 89 percent of the deflated sales by R&D-performing firms. Moreover, the coverage ratios are roughly constant over the entire sample period, varying only a few percentage points in each direction over two decades. Over this period, the matched sample also accounts for 68 percent of all successful U.S. patent applications by domestic non-government organizations (mostly corporations) and 73 percent of software patents granted to these organizations. These coverage ratios were also quite stable over the two decades.

However, only 37 percent of the R&D-performing firms contained in Compustat are matched to their patents in our data set and this coverage declined over the sample period as an increasing number of small firms have gone public since 1980. Thus this matched sample is broadly representative of the firms that perform most of the R&D and obtain the majority of patents, but it is not representative of entrants and very small firms. Nevertheless, given the extent of our coverage of R&D and patents, the results we obtain in our sample regarding patents and R&D will represent the overall interaction between patents and R&D.

It is possible that sample selection may bias regression coefficients within the matched sample. To check for this, we implement Heckman two-stage sample selection models (below) and find, in general, that sample selection has little effect.

D. Other data

We used the NBER Patent Citations Data File (Hall, Jaffe and Trajtenberg 2001a) to obtain data on citations received and numbers of claims. To obtain data on employment

[15] The original NBER sample accounted for 47% of the successful patent applications to U.S. non-government organizations; our sample accounts for 68% of these patents.

of programmers and engineers, we used the Occupational Employment Survey conducted by the BLS. This source provides detailed occupational employment of 3 digit SIC industries. Because not all industries were covered in all years of the survey during the early years, we linearly interpolated employment shares.[16] We also use a number of input price indices from the BLS Multifactor Productivity series.

II. Summary Statistics

Table 1 reports the number of software patents and other patents granted per year and also the numbers of applications per year, conditional on the applications successfully resulting in a grant by the end of 1999. As can be seen, their numbers have grown dramatically in absolute terms and also relative to other patents. Today almost 15 percent of all patents granted are software patents.

Table 1 also shows estimates of the number of software patents published by Greg Aharonian.[17] The overall trends are quite similar and the numbers in recent years are also quite close. Clearly, our definition of software patents is more inclusive, especially during the early years.

A. Who Owns Software Patents?

Table 2 shows characteristics of software patents compared with other patents, using data from the NBER patent database. Software patents are more likely to be owned by firms than by individuals or government. They are also more likely to be owned by U.S. assignees and to have U.S. inventors.[18] After the U.S., the top countries ranked by

[16] Thanks to Joseph Bush of BLS for providing the data. The employment categories we used for programmers were occupation codes 25102, systems analysts, and 25105, computer programmers; for engineers, we used 22100 (a group code). Because computer support occupations (25103 and 25104) were lumped in with the other codes during early years of the survey, we make a proportional adjustment in those years, reducing the employment counts of programmers by their relative share in the first year in which all categories were reported.

[17] Aharonian used the "I know one when I see one" criterion (private communication). For many years Aharonian has written articles on software patents in his *Internet Patent News Service* (www.bustpatents.com). The numbers cited in table 1 are reprinted from p. 319 of Lessig (2001).

[18] Allison and Lemley (2000) find that their sample of software patents has about the average share of U.S. inventors, although they use a somewhat different method to classify inventors' national origins.

inventors are Japan (18 percent), Germany (3 percent), Great Britain (2 percent), and Canada (2 percent).

Consistent with the findings of Allison and Lemley (2000), software patents tend to receive a larger number of subsequent citations, and they have substantially more claims per patent. Other research finds that these statistics may indicate greater *private* value, although not necessarily greater social value (Hall, Jaffe and Trajtenberg 2001b, Lanjouw and Schankerman 2001, Allison et al. 2003). In other aspects, software patents are similar to other patents.

To obtain more information about the firms that obtain software patents, Table 3 shows firm characteristics by patent type (software vs. non-software) for our matched sample. Relative to other patents, software patents tend to be obtained by firms with larger market value, sales, and R&D budgets. They are slightly more likely to be obtained by newly public firms. Allison and Lemley (2000) also find that software patents are more likely to be obtained by larger entities as classified by the patent office.

B. The Distribution of Software Patents Across Industries

Table 4 shows the industries of the firms obtaining software patents in the sample matched to Compustat. Most of the software patents are obtained by manufacturing firms, especially in the electronics and machinery industries, which include computers. Software publishers (SIC 7372) acquire only 5 percent of the patents in this sample, and other software service firms, excluding IBM, account for 2 percent.[19] The "usual suspects" for strategic patenting—SIC 35, 36, 38 and IBM—account for 68 percent of software patents.

The distribution of software patents across industries appears to reflect something other than the *creation* of software. Columns 2 and 3 include two measures that reflect software creation: the share of programmers and systems analysts employed in the industry and the share of programmers, systems analysts and engineers. These are the occupations most directly involved in the creation of software and so these shares

[19] IBM accounts for 13 percent of the software patents in our sample and it is consistently the largest software patentee. We break it out separately, as it is not representative of the software services industry.

represent the relative software development effort; the first measure more narrowly than the second.

The manufacturing sector acquires 75 percent of software patents but employs only 11 percent of programmers and analysts (32 percent of software writers if engineers are included). Software publishing and services (SIC 737, including IBM) acquires only 13 percent of software patents but employs 33 percent of programmers and analysts (18 percent if engineers are included). There is little reason to expect software developers employed in software companies or finance or retailing to be far less productive than software developers in manufacturing. These large differences suggest that industries differ dramatically in the degree to which they seek patent protection for their software.

This disparity also appears in the fifth column, which reports the differences in the simplest measure of patent propensity, the ratio of patents to R&D. Software publishing firms get only a quarter of the number of patents per dollar of R&D that other firms obtain. This corresponds to the views expressed by software publishing executives that software patents are of little value to them (USPTO 1994).[20] The sixth column displays a measure of software patent propensity derived from the regression analysis described below. Overall, software patents are more likely to be obtained by larger firms, established firms, U.S. firms, and firms in manufacturing (and IBM); they are less likely to be obtained by individuals, small firms, foreign firms, and software publishers.

III. The Rising Propensity to Patent Software

From 1987 to 1996 the number of successful software patent applications (granted by 1999) increased 16.0 percent per annum. This growth was greater than any of a number of yardsticks one might measure it against: during roughly the same period, real industrial R&D grew 4.4 percent per annum, employment in computer programming related occupations grew at a 7.1 percent rate, and real business spending on own-account and outsourced programming grew 7.4 percent per year. This growth occurred

[20] Also, BEA analysis of software investment (Parker and Grimm 2000) implies that about 30 percent of software is produced as packaged software, the primary product of firms in SIC 7372. Yet this industry acquires only 5 percent of software patents.

against a general background of rising patent propensity—the ratio of all domestic patent applications to real R&D—grew about 2.1 percent per year.[21]

We can identify several possible factors that might contribute to increased software patenting: growth in R&D generally, growth in that portion of R&D which uses software, greater productivity in software development, and changes in the cost-effectiveness of software patents from regulatory or other sources. To help sort out the roles played by these different factors, we use a "patent production function" model of Hall and Ziedonis (2001).[22] This production function relates the number of successful patent applications made by a firm each year to its size, relative R&D spending and other characteristics, plus a time dummy, which serves to capture residual changes in patent propensity.

A. Specification

In the preferred specification of this model, the expected number of patents for firm *i* in year *t*, conditional on firm characteristics for that firm and year, is

$$(1) \quad E[n_{it}] = \exp\left(\alpha_t + \beta_1 \ln Employees_{it} + \beta_2 \ln \frac{R\,\&\,D_{it}}{Employee_{it}} + \beta_3 \ln \frac{Capital_{it}}{Employee_{it}} + \beta_4 \delta_i \right)$$

where δ is a dummy variable that equals one if the firm is a new entrant and zero otherwise. The right hand side variables capture the effects of scale, R&D intensity, capital intensity and new entrant status. The time dummy then captures changes in the propensity to patent. Differences in the time dummies between two different years correspond to log differences in the expected number of patents. This is our initial specification to which we add additional controls, including firm effects and a measure of software development intensity.

---

[21] Total industrial R&D increased from $121 billion in 1988 to $164 billion in 1998 in 1996 dollars (NSF 2003). The BLS Occupational Employment Survey estimates 904,430 employees in "computer scientists and related occupations" in 1987-89 and 1,839,760 in 1998. The BEA estimates $38billion in 1996 dollars for business spending on own-account and custom software in 1989 and $64billion in 1998 (Parker and Grimm 2000). The general increase in patent propensity has been explored by Kortum and Lerner (1999) and Hall and Ziedonis (2001).

[22] The literature on patent production functions also includes Scherer 1965, Bound et al. 1984, Pakes and Griliches 1984, and Griliches, Hall, and Hausman 1986.

Since the left hand side variable is a count and many observations are zero, the equation can be estimated with a Poisson regression, as is frequently done in the literature. The Poisson regression assumes that the variance equals the expected value of the left hand side variable, but often Poisson specifications fail to meet this assumption and show "over-dispersion." Tests using a negative binomial specification reveal over-dispersion in our data too, and, following Hall and Ziedonis (p. 113), we take this as an indication to use heteroscedastic-consistent standard errors.

Our analysis differs from Hall and Ziedonis in two important ways, however. First, our dependent variable is not all of the patent applications of the firm, just software patent applications. In principle, this should cause no problem—one still expects size, R&D intensity, etc. to affect the level of software patents. But we may also want to control for the degree to which the firm directs resources to software development, which we do below. Second, Hall and Ziedonis study a narrowly defined industry whereas we study a broad range of industries. For this reason, we will want to control for industry or firm effects in some of our regressions.

In our base specification n_{it} is the number of software patents applications by firm i in year t that resulted in a patent granted by 1999.[23] Because patent prosecution typically takes two years or so, we conduct our regressions through 1997. The other variables are as follows: employees is the number of employees listed in Compustat in thousands, R&D is deflated by the GDP deflator, and capital is property, plant and equipment deflated by the NIPA capital goods deflator. The "new firm" dummy is equal to one for the first five years a firm appears in Compustat.

B. Results

Column 1 shows the base regression. Column 2 adds seven industry dummies and variables to capture the relative use of programming personnel. The first of these variables is the ratio of "computer scientists and related occupations" to total industry employment in the BLS Occupational Employment Survey (OES) for the firm's SIC 3

digit industry (or 2 digit if Compustat assigns the firm only two digits). The second variable is the comparable share of engineers in total employment. The OES was conducted on a three year cycle until 1995, so values for this variable in intervening years are linearly interpolated. And for comparability, we only used OES data from 1987 onwards, so the second column covers a shorter interval.

The elasticity of the scale variable, employment, is similar to estimates in previous studies. Hausman, Hall and Griliches (1984) obtained an elasticity of .87 on R&D. Hall and Ziedonis obtained an elasticity of employment of .85 when they included some firm controls. The coefficient of the R&D intensity variable is, however, much higher than in Hall and Ziedonis, although when we include firm fixed effects below, this difference largely disappears, suggesting that it is picking up unmeasured firm/industry heterogeneity. Capital intensity (in column 2) is also significant and similar in magnitude to one estimate by Hall and Ziedonis, but smaller than another. Hall and Ziedonis interpret this coefficient as evidence that capital intensive firms may patent more because they are subject to holdup by rivals who patent strategically. That is, this is evidence of "defensive patenting." Our result suggests this hypothesis might apply more generally.

1. New vs. Old Firms

We find that the "new firm" dummy variable is not significant. In contrast, Hall and Ziedonis find that their dummy variable is highly significant and has a relatively large coefficient.[24] They suggest that their result arises from new semiconductor design firms that need patents to secure financing (see also Hall 2003). These smaller firms do not have complementary manufacturing facilities that may provide another means of appropriability.

---

[23] In our data, we only observe patent applications that are successful. After November 2000, patent applications are generally published 18 months after the filing date. But publication is not required if the inventor does not seek patents abroad.

[24] There is also a difference between the definition of our new firm variable and theirs. Hall and Ziedonis include all firms that entered Compustat after 1982. We include firms that entered in 1982 or later, but our dummy variable equals one only for the first five years of entry. Their initial interest was to compare whether incumbent firms in particular benefited from the creation of the Court of Appeals for the Federal Circuit. Our interest is whether entrants appear to find additional benefits from software patents, something Hall and Ziedonis explored specifically for entrant design firms with an additional dummy variable.

Although our sample may not be representative of the entire population of new firms, this regression does include a reasonably large sample of new firms (1,308 observations during the first five years of 314 firms). Hence our different result suggests that either software patents are not comparably useful for obtaining financing in general or this "vertical disintegration" strategy is not broadly relevant outside the semiconductor industry, or both.

We explored this issue further by interacting the new firm dummy variable with industry dummy variables. Only one of these interaction terms was statistically significant, the one for SIC 73, business services, which in our sample largely consists of software services and pre-packaged software firms. Column 3 shows a regression with just this interaction term added. The coefficient is negative and statistically significant, suggesting that new software firms obtain *fewer* software patents than established firms.[25] This suggests that new software firms may not obtain the same benefit from patents as do new semiconductor firms.

2. Cross Industry Variation in Software Patent Propensity

Returning to Column 2, the coefficients on the industry dummies indicate large inter-industry differences in software patent propensity even after controlling for industry employment of programmers and engineers. Given the exponential specification, (1), differences between industry coefficients correspond to differences in the log of software patent propensity. Column 6 of Table 4 shows corresponding differences in software patent propensity itself (that is, the exponential of the coefficients), normalized so that the software patent propensity of SIC 73 equals one (this includes IBM). As can be seen, the differences are quite large, with SIC 36 and SIC 38 obtaining an order of magnitude more software patents, all else equal, and machinery, SIC 35, not too far behind. In general, the industries that have a high propensity to patent software also have a high patent propensity in general (see column 5 of Table 4), although the differences in

[25] We also ran these regressions using the total number of patents, not just software patents, as the dependent variable and obtained similar results (not shown). Graham and Mowery (2003) report that the software patent propensities of established software publishers rose over the 1990s, but for entrant firms (those founded after 1984) there was no discernable trend.

software patent propensity are larger. In the literature, these are the industries known for strategic patenting (see section IV).

The magnitude of these differences and the known importance of strategic patenting in electronics and computers, suggests that these results may be the outcome of strategic patenting. To test this idea further, column 4 drops the industry dummies but includes instead a measure of the degree to which other firms in the industry patent. This variable is the number of all patents obtained by other firms in the same 2 digit industry as the observed firm, divided by the employment of those firms. The positive and significant coefficient suggests that firms obtain more software patents in industries that patent more overall, all else equal. This result is consistent with "defensive" patenting, although it could also arise from other industry differences, such as large differences in alternative means of appropriability.

3. Unobserved Heterogeneity

Our seven industry dummy variables do not capture the full extent of inter-industry heterogeneity or other firm heterogeneity. This was confirmed by tests using the random effects and fixed effects Poisson models in Hausman, Hall, and Griliches (1984). We then face the choice of better controlling for this unobserved heterogeneity via a random effects or a fixed effects model. A Hausman test rejects the null hypothesis that the random effects estimates are consistent; that is, the firm effects appear to be correlated with the coefficients (P = 0.000), indicating that fixed effects are preferred.

Column 5 presents the fixed effects Poisson regression. This is a conditional maximum likelihood regression where the likelihood of each observation is conditioned on the likelihood of the observed sum of software patents for each firm over all years in the panel. This is only calculated for firms that have software patents in at least one year. Since many firms obtain no software patents, the sample size is reduced considerably for this regression. With fixed effects, the scale elasticity is somewhat smaller, and is consistent with earlier research (Hausman, Hall, and Griliches 1984) and the R&D intensity coefficient is now much more in line with the estimates in Hall and Ziedonis (2001).

In Figure 1, we plot the year dummies from this regression (normalized to equal 1 in 1987), those from the Column 1 regression, which spans a longer time period, and the corresponding year dummies from Hall and Ziedonis's preferred specification. As can be seen, after the mid-80s, all three series grow rapidly, persistently and at about the same rate. Compared to the rate for 1987, and holding all other factors constant, firms were successfully applying for nearly 50 percent more software patents in 1991, and 164 percent more by 1996.

4. The Ebb and Flow of Copyright Protection

One factor that may explain the large increase in software patent propensity is diminishing protection for computer programs provided by copyright. Lemley and O'Brien (1997) describe the rise and fall of the "nonliteral infringement" doctrine. Established in the 1986 decision in *Whelan Associates v. Jaslow Dental Laboratories*, it gave copyright holders some protection over the features of their software programs. But the doctrine was rejected in the 1992 decision in *Computer Associates International v. Altai*.[26] If the alternative of copyright protection affected firms' propensity to obtain patents, then one might expect a decrease in patent propensity after 1986, and then an increase after 1992. No significant fluctuation is observed in Figure 1 and regression tests for a break find only a very small change in trend.[27] Moreover, the industries that patent most heavily also tend to use embedded software, so copyright protection was less relevant to them.

C. Interpretation

We use the fixed effects specification to interpret the relative influence of various right hand variables. Because (1) is exponential in form, changes in variables times their coefficients affect the log of software patents and can be interpreted as growth rates. To calculate annual growth rates, we evaluate the mean of each variable in 1996 and subtract

[26] 797 F.2d 1222 (3rd Cir 1986) and 982 F.2d 693 (2d Cir 1992), respectively.

[27] We used a regression similar to column 5 of Table 5 with a time trend instead of year dummies. We interacted the time trend with an interval dummy for the years 1986-92 and found a very small (-.002), but statistically significant reduction in the rate of growth in software patent propensity during this interval. This effect is on the order of 5% of the overall growth rate in software patent propensity.

the mean in 1987. We then multiply this difference by the estimated coefficient from column 5 and divide by 9 (the number of years elapsed) to obtain an annual contribution of the variable to the growth of software patents for our sample. The resulting estimates are shown to the right of column 5. A similar calculation for the year dummies is shown for all columns in a separate row.

The annual growth of software patenting among the firms in this sample (which excludes firms without any software patents over this entire period) is 16.4 percent, slightly larger than the aggregate 16.0 percent reported above. The majority of this increase is captured by the contribution of the year dummies (10.8 percent). The next largest contribution is from greater capital intensity (1.8 percent) followed by the growth in programmer employment (1.2 percent), engineering employment (1.2 percent) and R&D intensity (1.1 percent).

Thus the majority of the growth in software patenting is not attributed to any of these explicit controls and can be attributed, instead, to rapidly rising patent propensity. Note that this increase in patent propensity is quite close to the result obtained by Hall and Ziedonis (2001) for all patents in just the semiconductor industry. A comparable calculation on their preferred specification also results in a 10.8 percent annual growth in patent propensity from 1987 – 95.[28] But such growth rates are far from typical for total patenting in most industries.

1. Taking into Account Productivity Growth

In principal, software patent propensity can be decomposed into two factors: an increase in the productivity of software developers, and change in the cost-effectiveness of patenting.[29] The first factor means that software developers may produce more inventions using the same level of inputs. The second means that firms find it more attractive to patent a higher proportion of inventions, perhaps because the cost of doing so has fallen, or because such patents provide larger benefits than they did in years past, or possibly both.

[28] Thanks to Rosemarie Ziedonis for graciously sharing data.
[29] See Hall and Ziedonis (2001) for a similar discussion.

A thumbnail calculation suggests that changing cost-effectiveness is the major factor accounting for the rise in patent propensity. The productivity of software developers is notoriously difficult to estimate. Although there has been strong market for software development tools, the actual contribution of these tools to productivity is widely debated and studies find that computer aided software engineering tools often go unused (Kemerer 1992). Nevertheless, we can calculate an upper bound for programmer productivity growth using price indices for the pre-packaged software industry. As we discuss below, these estimates are likely to exaggerate productivity growth for software developers involved in the R&D process.

From 1992 to 1997 receipts of the pre-packaged software industry (SIC 7372) grew 18.5 percent per annum while employment grew 14.2 percent per annum (Census 2003). From the OES survey, we know that within SIC 737, employment of computer science related occupations grew 2.4 percent faster than total employment, so we estimate that programmer employment in pre-packaged software grew 16.6 percent per annum. Thus nominal growth of software revenue per programmer was about 1.9 percent per annum, less than a fifth of the estimated increase in software patent propensity.

But we must also take into account improvements in the quality of software produced. A recent literature has developed price indices for pre-packaged software using both matched-model methods and hedonic methods (see Grimm and Parker 2000, and Abel, Berndt, and White 2003). The BEA has a matched model estimate of a -3.5 percent price change per annum over this period and this corresponds well to other matched model estimates. Hedonic price estimates, based on a small number of applications, usually show more rapid price declines, although some economists have questioned whether these estimates are reliable (Oliner and Sichel 1994). The BEA makes a seat-of-the-pants quality adjustment to come up with a pre-packaged software price index that has a growth rate of -6.8 percent over this same period. Combining these estimates, real pre-packaged software per programmer grew either 5.4 percent per annum (using the matched model index) or 8.7 percent per annum (using the quality adjusted index).

These figures are clearly less than the 10.8 percent annual rate of patent propensity growth. But they very likely substantially overstate the growth in programmer

productivity *per se*. The BEA, in fact, attributes *most* of estimated price decline to growth in the market for software driven by rapidly falling prices for complementary computer hardware (Grimm and Parker 2000, p. 6). With the total market growing rapidly each year and small marginal costs of production, much of the estimated price decline is attributable to economies of scale. But what matters for patent propensity is the productivity of software developers at actually creating new software (e.g., new titles of pre-packaged software) rather than duplicating software that already exists. Subtracting the large scale economies, this productivity growth rate must be substantially less than 5.4 percent or 8.7 percent per annum. At the very least, growing cost-effectiveness of patents accounts for some substantial portion of the rise in software patent propensity and may well account for most of it.

D. The Cost-Effectiveness of Software Patents

Software patents may have become more cost-effective because of the regulatory changes described in Section I. Eliminating the subject matter exclusion and reducing the non-obviousness and enablement requirements may have made software patents much easier (less costly) to obtain. Stronger enforcement and a greater presumption of validity may have increased the appropriability each patent delivered. Both served to decrease the cost of appropriability.[30]

Strategic patenting may have amplified the effect of these changes. Reductions in the cost of appropriability may encourage firms to pursue more aggressive strategic behavior (Bessen 2003). This may, in turn, induce other firms to engage in defensive patenting, further increasing the patent propensity. Both the importance of capital intensity and the large industry effects found above suggest that strategic patenting may play an important role and that this role may have increased over time.

In summary, the outsized growth in software patenting is not adequately explained by research inputs, productivity growth, or other observable factors. There is a

[30] Another possibility is that alternatives to patents became less effective, increasing the relative cost effectiveness of software patents. But survey evidence suggests that trade secrecy did not decrease in importance and may have increased (Cohen, Nelson, and Walsh 2000). We address changes in the efficacy of copyright protection in section III.

significant residual trend in software patent propensity on the order of 5 percent to 10 percent a year, roughly 2.5 to 4 times larger than the trend for patents in general. Such increases would be consistent with firm responses to regulatory changes that have increased the appropriability of software patents, made them easier to obtain, or both. Moreover, the pattern of large inter-industry differences in patent propensity (and the importance of capital intensity) are consistent with models of strategic patenting

IV. R&D and Software Patents

The traditional incentive theory suggests that more cost-effective patents for software should lead to increased R&D spending, all else equal. With "cheaper" and/or "stronger" patent protection, firms should patent more software inventions and/or realize greater profits from the software inventions they patent. Expecting greater profits, firms should find it profitable to spend more on R&D, all else equal. If the incentive hypothesis is correct, the increase in the cost effectiveness of software patents identified in the preceding section should be positively correlated with increases in firms' R&D investments. In this section, we test this hypothesis.

To do so, we must control for other factors so that, to the extent possible, we hold "all else equal." In a dynamic world with shifting demand and supply and changing prices, firms may increase or decrease their R&D spending for reasons unrelated to changes in the appropriability of software patents. For example, a firm facing growing demand may increase R&D because increases in demand may make heretofore marginally unprofitable R&D projects profitable. This renders any simple correlation between software patenting and R&D spending unpersuasive.

Of course empirical economists have developed techniques for analyzing the relationship between two quantities against a backdrop of shifting prices, namely, by estimating elasticities of factor substitution (see Berndt 1991 for a general review). These techniques allow economists to identify whether two factors are complements or substitutes. The problem here is quite similar if one considers R&D and patents to be factors of production (for the production of profits, rather than physical output). In other words, the incentive theory is akin to saying that patents and R&D are *complements*.

Greater use of patents should be associated with greater use of R&D. And if patents become more cost effective—that is, if their quality-adjusted price falls—then the share of resources allocated to R&D relative to other factors of production should increase.

A. A Simple Model

To see that the incentive argument is equivalent to complementarity of R&D and patents, it is helpful to consider a simple generalization of the formal patent race models that have been used as the basis of the incentive hypothesis. In these models, the more R&D a firm does on a project, the greater the probability that the firm will make a successful product innovation (first). Abstracting away from issues of timing and taking the actions of other firms as given, let $P(r)$ be this probability of success. Since R&D activities on this project will have diminishing returns, we assume that P is increasing, concave and that $P(0) = 0$. Let r be the quantity of R&D performed at price w.

If the firm successfully innovates and gets a patent, then it earns a discounted stream of rents during the term of the patent (and perhaps some profits afterwards). If the discounted stream of production costs associated with the product is C, then the profits equal $A \cdot C$, where A is the *markup above cost*. The expected profits on the project are

$$\pi = A \cdot C \cdot P(r) - wr. \tag{2}$$

In the patent literature, policy features such as patent term and scope affect expected profits, and these features correspond to different levels of A. In other words, A can be interpreted as a measure of appropriability.[31] To the extent that a project can be protected by software patents, stronger, more cost-effective patents should increase A. The firm's optimal level of r can be found by examining the first order condition of (2). Then straightforward calculation shows that increases in A generate increases in the equilibrium level of r. Moreover, if the scale of production remains relatively constant, then the ratio of R&D expenses, $w \cdot r$, to production cost, C, will also increase.

It is theoretically possible, however, that the scale of production for a successful project may increase or decrease with appropriability—for example, an increase in A

[31] See Arora, Ceccagnoli, and Cohen (2003) for a similar decomposition taken to the data.

might be associated with a change in market structure from a duopoly to a monopoly. Diverse empirical evidence suggests that substantial changes in production scale are unlikely in the broad range of industries studied here. Cohen and Klepper (1996) argue that "the level of output over which rents from R&D are realized is closely related to the firm's output at the time it conducts its R&D." More generally, in many industries, firms achieve appropriability mainly through lead time, learning-by-doing, complementary services, etc. and patents only play a secondary role (Levin et al. 1987, Cohen, Nelson, and Walsh 2000). Any improvement in appropriability conveyed by patents is unlikely to have a drastic effect on market structure in these industries because firms *already* have significant appropriability. In Table 6 below, we find that software patents did not appear to influence profit margins significantly, also suggesting that industry structure is unlikely to change dramatically in response to the availability of software patents.

A positive association between appropriability and R&D cost share is a necessary condition for patents and R&D to be Hicks-Allen complements. It also fits nicely with a standard technique for evaluating factor demand by using cost share equations. This technique uses relative cost shares (the cost of an input factor relative to total production costs) regressed against log relative prices of input factors. For example, the ratio of energy costs to total production costs is regressed against log relative prices of labor, capital, energy, etc. These equations are derived from a flexible form translog cost function (see Greene, 1997, Chapter 15.6). For a regression that uses one input factor as a dependent variable (say, energy), if the relative price of another factor (say, labor) has a positive coefficient, then these two goods cannot be complements.[32] Moreover, this test takes into account both the direct effect of price changes on consumption of the left hand side good (the substitution effect) and also the overall effect of the price change on total costs (the income effect).

B. Estimation

To apply this technique to our problem, we can treat both R&D and patents as input factors in addition to conventional input factors. Then, following the standard analysis, a

[32] To be complements, the coefficient must be less than $(-1) *$ the product of the two cost shares.

regression equation can be derived from a procedure of minimizing costs while holding revenues constant. This equation would be

$$(3) \qquad \frac{r}{C} = \alpha + \gamma_0 \cdot \ln c + \sum_j \gamma_j \cdot \ln p^j$$

where C is total cost (redefined here to include R&D costs as well as production costs), c is the "cost of appropriability" (the quality-adjusted cost of patents, not observed), p^j are prices of other input factors (taking one as numeraire), and the Greek symbols are coefficients to be estimated.[33] This technique is preferred to estimating quantities when analyzing firm-level data under the assumption that prices are likely to be exogenous to the firm (Berndt 1991).

Since we do not observe the cost of appropriability, we do the following: We include time dummies in the regression which will soak up, among other things, overall changes in the cost of patenting and the "strength" of patents. We believe that a substantial part of the rise in software patenting is a response to increases in the "strength" of patents for software inventions and/or decreases in the cost of patenting these inventions (see section III). In that case, the share of software patents in a firm's total patenting, denoted s, serves as a proxy that captures variations in the relative cost of appropriability across firms and over time.

Taking differences to sweep out firm fixed effects, our regression equation is

$$(4) \qquad \Delta \frac{r_{it}}{C_{it}} = \alpha_t + \beta \cdot \Delta s_{it} + \sum_j \gamma_j \cdot \Delta \ln p_{it}^j \ .$$

We use five-year differences to reduce noise from measurement error and any biases associated with partial factor adjustment.

We calculate the R&D cost share as the ratio of R&D spending to total costs measured as the sum of cost of goods sold and selling, general and administrative

[33] Typically, a system of equations is jointly estimated for each input factor (but one). Here, we estimate only a single equation, in differences, because we do not have full information on other factor inputs. Our estimates are still consistent, but not as efficient as they would be if we could estimate multiple equations.

expenses. Since not all firms report these cost items, we also use an alternative measure, R&D/sales (see below).

The software share of patents, *s*, is measured as the number of software patents applied for in a given year divided by the total number of patents applied for that year, conditional on the patents having been issued by 1999. Since patents take on average about two years to issue, we only use patent applications through 1997. Also, this software patent share measure is subject to sampling error—this measure will have large variance for firms with small numbers of patents. To reduce heteroscedasticity, we use Weighted Least Squares and we also report heteroscedastic consistent standard errors.[34] Below, we check that our results hold without this weighting. We also check for possible correlation between Δs and the error term.

The price variables are annual industry price indices from the BLS Multifactor Productivity report at the two-digit level for manufacturing and for the private non-farm business sector for non-manufacturing firms. In addition to capital, labor, materials, energy, and purchased services, we include an index for the price of IT capital, to control for the influence of IT.[35] To check that our results did not depend on any possible errors in these prices, we also ran our main regressions using a full set of interacted year and two-digit industry dummies. The coefficients of interest changed only slightly.

C. Results

Column 1 in Table 6 shows a regression of equation (4) for the years 1991-97. The coefficient on software patent share is negative and significant at the 5 percent level. A negative value indicates a negative conditional correlation between our measure of the declining relative cost of appropriability and the R&D cost share. That suggests these two inputs are substitutes and not complements.

---

[34] The weight used is $\left(1/n_t + 1/n_{t-5}\right)^{-1}$, since the sampling variance for *s* is proportional to *n*.

[35] We use data from the April 2001 release. Thanks to Bill Gullickson and Steve Rosenthal of BLS for providing the data. We also used separate indices for the components of IT capital, but these did not change the non-price coefficients and they were highly multi-collinear.

As noted above, the cost variable is missing for many observations. Firms reporting cost of goods sold tend to fall disproportionately in manufacturing and this might bias results. Column 2 tests the significance of this sample selection and also of firms not matched to the patent file by running a Heckman sample selection model. This model consists of a regression equation and a probit equation (not shown) estimated on a binary variable indicating whether the observation is included in the regression reported in column 1. The regression coefficients are adjusted to reflect biases arising from the sample selection process. For the probit, we used log market value, log deflated sales, a new firm flag and year dummies on the right hand side. The probit coefficients are highly significant and a likelihood ratio test indicates the probit regression as a whole is highly significant. While a Wald test rejects the null hypothesis that the two equations are independent, any resulting bias in the coefficient of software patent share seems small. Estimated with the sample selection correction, the coefficient remains negative and is highly significant.

Another way to test the incentive hypothesis is to see whether software share changed the markup over cost. To do this, column 3 regresses the change in log deflated sales against the change in software share and the change in log deflated cost. The coefficient of the change in log cost is not significantly different from 1, consistent with constant returns to scale. If software patents increased profit margins, then in this regression the software share should have a positive coefficient. The estimated coefficient is positive but quite small and not statistically significant. It appears the software patents did not have any substantial effect on markups.

Although sample selection does not seem to indicate a substantial bias, our regressions will be more representative, and estimated more precisely, if we expand the sample size. We can do this by using the ratio of R&D to sales—R&D intensity—as the dependent variable instead of R&D cost share. So long as firm profit margins do not change dramatically, the ratio of R&D to sales will remain roughly proportional to the ratio of R&D to cost. We verified that the ratio of cost to sales does not change much on average (there is a variance of about .001 per year) where cost data is not missing. Also, the previous regression shows that changes in profit margin are uncorrelated with

changes in software share. This means that using R&D intensity instead of R&D cost share will not bias the coefficient of software share. Finally, to eliminate cases where profit margins were likely to change dramatically, we trimmed a small number of observations where R&D spending exceeded half of annual sales.[36]

Column 4 of Table 6 repeats the basic regression equation using the change in R&D intensity as the dependent variable. The sample size is almost four times larger and standard errors are about one third the size of those in column 1. The coefficient on software share is quite similar. Further, in column 5 we repeat the Heckman sample selection model regression. Here the null hypothesis that the sample selection equation is independent of the regression equation cannot be rejected. Thus using the change in R&D intensity as the dependent variable appears to provide a more representative and larger sample. For this reason, we use this dependent variable in the subsequent analysis.[37]

Our historical and legal research suggests that the substitution of patents for R&D might be explained by changes in the cost/benefit of software patents—that these patents were made "stronger" and/or cheaper during the 90s. Consistent with this view, we conducted the previous regressions from 1991 to 1997. To check that this time period is not arbitrary, column 6 repeats the regression of column 3 over a longer period with the change in software share interacted with three time dummies. These results show that the substitution effect follows the same time pattern as was observed for patent propensity in the last section. There was a positive, but insignificant effect during the late 80s, and negative and highly significant coefficients during the 90s. The magnitude of the substitution effect appears to increase in the late 90s, but the difference from the early 90s is not statistically significant.

The observed substitution effect is economically significant. Thumbnail calculations suggest that by the end of the 1990s, R&D would have been about 10

[36] This screen eliminated 121 observations in total of some 39 firms. These firms were mainly startups (34), mostly biotech (25) and each of these firms had observations for other years that were included in the sample. To verify that this trimming did not introduce a bias, we performed a Heckman sample selection analysis on just the selection created by this trimming. The null hypothesis that the sample selection equation (with log market value, log sales, new firm flag, R&D intensity, lagged R&D intensity, and year dummies as regressors) was independent of the regression equation could not be rejected (P = .859).

percent higher without the substitution. This is equivalent to about $16 billion in private R&D, or roughly eight years of the long run average increase in R&D intensity among American firms.[38]

1. Robustness Checks

To verify that our results were not unduly influenced by the use of weighted least squares, we repeated the regression in column 4 using ordinary least squares but eliminated observations for years where a firm filed fewer than five patents. This left 1,933 observations and a coefficient estimate for Δs of -.010 (.005), suggesting that the substitution effect exists without the weights, but WLS estimates are more efficient.

It is possible that Δs might be correlated with the error term because R&D and patenting decisions are made simultaneously. To test this, we instrumented Δs with lagged values and re-ran the regression in column 4 as an IV regression. When we instrumented with one and two year lags, the resulting coefficient was -.103 (.033), suggesting that any bias might actually be downwards. Comparing the instrumented and non-instrumented regressions, a Hausman test could not reject the null hypothesis that the uninstrumented version is consistent with a P value of .999 ($\chi^2(13) = 1.74$). So endogeneity of Δs does not appear to be a problem. Similar results were obtained using just a two-year lag.

To make sure that the substitution result was not influenced by possible error in the BLS price series we used, we replaced the changes in log prices with terms for the year dummies interacted with 2 digit industry dummies.[39] Thus, we picked up all industry trend variables in addition to sweeping out individual firm effects through differencing. Given the large industry effects shown in Table 5, undoubtedly some of the variation in Δs was picked up in these interaction terms, reducing the coefficient on Δs. As expected,

---

[37] We also performed all of the following analysis using the change in R&D cost share as the dependent variable. Results were generally quite similar, but with larger standard errors

[38] A 10 percent increase in software share over the decade times a coefficient of -.037 divided by mean R&D intensity of about 3.5 percent. The dollar estimate is based on the level of industrial R&D in 1998 (NSF 2003). The average annual increase in the ratio of industrial R&D to net sales, as reported by NSF, is .04 percentage points.

[39] Several industries had very few firms, so we combined seven 2 digit industries into other industries.

the estimated coefficient was a bit less, but still significant at the 1 percent level, -.025 (.010). Thus the substitution effect appears robust to a variety of estimation concerns.

2. Industry

Column 1 of Table 7 considers whether our results may be limited to one or two industries by interacting the change in software share of patents with industry dummies. We find a significant negative substitution effect for SIC 35 (which includes the computer industry), SIC 36 (electronics, including semiconductors), and SIC 73 (business services, including software and IBM).[40] For other industries, the coefficient is not significantly different from zero.

The substitution effect thus seems to occur mainly in those industries known for strategic patenting. Column 2 divides the industries into strategic ones (SIC 357, 36, 38 and IBM) and the rest. Firms in the strategic patenting industries exhibit a large and highly significant substitution effect; for firms in other industries, the coefficient is not significantly different from zero, indicating neither a substantial substitution nor complementary effect.

3. Outsourcing Software Production

The use of software throughout the economy grew rapidly during the 1990s and so it is important to examine the possibility that the increasing ubiquity of software may somehow spuriously cause the correlations observed in Table 6. There are two potential mechanisms to consider: the use of purchased software within the firm generally, and the use of software in the R&D process itself.

Investment in software also grew rapidly during the 90s, especially in the form of purchases of pre-packaged software (see Grimm and Parker 2000). It is possible that as firms purchased more software from other companies, they reduced the internal

[40] If IBM is excluded, the coefficient on SIC 73 is no longer significantly different from zero.

production of software, some of which may have been classified in the R&D budget.[41] This would have reduced R&D spending and R&D intensity.

Two arguments militate against this explanation. First, one would expect that if firms are developing less software internally, they are less likely to obtain software patents. In that case, software patent share would not rise as much, or might even fall. Then the correlation between changes in software patenting share and R&D intensity would likely be *positive*. Second, our regressions control for externally purchased IT, which includes software (both outsourced and pre-packaged).[42] To test this further, we also performed the regressions with separate price indices for computers and software (these are components of the BLS IT price index) and found essentially the same results. So the use of purchased or outsourced software does not seem to explain the negative association between software patent share and R&D intensity reported in Table 6.

4. The Use of Software in R&D

The second possible mechanism for a spurious correlation is an increase in the importance of software in the R&D process itself. Firms employ software to design products, both by using tools, like computer-aided engineering software, and by using embedded software in hardware products that were formerly hard-wired. In other words, software may reduce the cost of performing R&D. So in principle, it is possible that firms that use software to design new products might obtain patents on this software and, under certain conditions, they might also spend less on R&D.

There are several reasons to suggest that this does not explain the negative correlation we observe. First, this explanation only works if the demand for R&D is relatively inelastic. If software use reduces the cost of performing R&D, then R&D projects that were marginally unprofitable before become profitable. If there are a sufficient number of marginal projects, then the lower effective cost of each project will

[41] In Compustat, software development spending is included in R&D, unless it is paid for by a customer.

[42] The coefficient on the change in the log of the IT price index is consistently positive, indicating that IT does substitute for R&D. Dropping this variable from the regression only changes the coefficient of *s* slightly. That suggests the relationship between utilization of IT and R&D is different from the interaction between software patents and R&D.

be offset by the increase in the number of projects performed. That is, if R&D demand is sufficiently elastic, any reductions in the effective cost of conducting R&D would cause firms to *increase* R&D intensity. We know from studies of the R&D tax credit that the demand for R&D is rather elastic with respect to its tax price. Berger (1993) finds that reductions in the tax price of R&D *increased* R&D intensity. In a survey of the literature, Hall and van Reenen (2000) find that the tax price elasticity of R&D is around -1; at this level, changes in the price of R&D should have little effect on R&D intensity. Thus, this evidence suggests that the demand for R&D is sufficiently elastic so that software use cannot explain the negative correlation.

Second, we can also control for software use. If increased use of software in the R&D process is causing the negative coefficient on the change in software share, then including the change in the relative employment of programmers and engineers in the regression should reduce or eliminate this association. Column 3 of Table 7 repeats the R&D intensity regression, but includes the change in the share of programmers and systems analysts in industry employment and the change in the share of employment of engineers. The coefficient of Δs hardly changes, indicating that the substitution effect is orthogonal to the employment of software writers.[43]

Third, it is hard to reconcile this explanation with the cross-industry pattern of the substitution effect. Software is used in the design and development of new products and processes in a wide range of industries, some of them strategic, such as computers and electronic equipment, and some in the non-strategic group, such as automotive, chemical processes and non-computer machinery. But although both groups of industries use software in products and in the design of products, the substitution effect is observed only in the strategic group. It seems unlikely this pattern is simply coincidence.

[43] The coefficient for programmers is negative and significant. One interpretation of this is that firms shifting from manufacturing to services may reduce their R&D and increase their IT resources, resulting in an increase in the employment share of programmers. The mean change in programmer employment share is .001, so the net effect is not very large. Moreover, using interaction regressions, we found that this effect is largely limited to non-manufacturing industries.

5. Variation by Firm Size, Incumbency, and Other Factors

Some researchers suggest that stronger patent protection for software may facilitate vertical dis-integration. In particular, small firms may have increased their software R&D in order to acquire more patents to license or sell to large firms. If this were so, software patents might complement R&D at small firms. On the other hand, if software patents are associated with strategic patenting, then small firms might be forced to acquire "defensive" patents and they might also face lower appropriability. Then R&D would substitute for software patents at small firms.

Column 4 explores the effect of firm size by interacting the change in software share with dummies for firms of small and medium employment size (less than or equal to 1,000 employees and greater than 1,000 but less than or equal to 10,000).[44] The coefficients on these interaction terms thus indicate differences between the smaller size classes and large firms. Large firms exhibit the strongest substitution effect, but the effects for the smaller classes are not significantly different statistically. And the smallest size class hardly differs from the largest class. These results do not support the view that the substitution is purely a large firm effect or that software patents have encouraged greater R&D among small firms.

Column 5 looks at the related question of differences between incumbent and entrant firms. It adds an interaction of the change in software share with a dummy variable that is 1 if the firm entered Compustat after 1985. There were 297 observations where this dummy was set in this regression. Here again, the substitution effect for entrants differs only slightly from the effect for incumbents and this difference is not statistically significant. We repeated the regressions in Columns 3 to 5 for just the group of firms in the strategic patenting industries and obtained similar results.

We also conducted other regressions (not shown), which included additional controls for firm size (lagged log deflated sales), new firms, firm risk (standard deviation of annual stock price growth), and liquidity (change in long-term debt divided by capital). Including these control variables had little effect on the coefficient on software

share. Finally, we considered the role of accounting changes but concluded this could not explain the observed relationship between software patents and R&D intensity.[45]

V. Interpreting Our Results

While our empirical technique cannot identify the causal relationships, it does reveal a number of patterns we can compare to a variety of theories about the effects of extending patent protection to computer programs. For example, the facts seem difficult to reconcile with the traditional incentive hypothesis.[46]

Our results make quite clear that software patenting by and large has little to do with the pre-packaged software industry. That industry acquires a very small fraction of all software patents. Although it has been argued that software patents may encourage entry into an industry dominated by powerful firms, software entrants have a relatively low software patent propensity. However, we do not have a comprehensive picture of all entrant firms.

Two-thirds of all software patents are obtained by a small group of industries known for strategic patenting (SIC 35, 36, 38 and IBM). Moreover, this group appears to play a unique role throughout our analysis. This group has a much higher propensity to obtain software patents and, in the 1990s, appears to have substituted these patents for R&D. For firms in other industries, there does not appear to be any relationship between an increasing focus on software patenting and R&D investments.

We think the central role of this group of industries is not accidental—that is, the phenomenon we observe with software patents may be the *same phenomenon* other researchers have observed for patenting in general for these industries:

---

[44] Of the 2,991 observations, 432 have employment less than 1,000 and 1,177 have employment greater than 10,000.

[45] Beginning in 1985, the FASB required firms to capitalize software development expenses (but not research or maintenance expense, which usually account for most software costs). Based on some simple calculations (details available from authors) we find that this change can explain neither the timing nor the sign of our results.

[46] In this discussion it is important to recall that the estimated effects are based on patentability standards as they existed after the early 1980s. It is possible the effects might have been different under different (say higher) patentability standards, but that question cannot be addressed here.

1. Firms in these industries report that patents are relatively less effective compared to firms in other industries, yet they obtain patents to aid negotiations, to cross-license, to block competitors, and to prevent suits (Levin et al. 1987, Cohen, Nelson, and Walsh 2000). Firms in some of these industries engage in strategic cross-licensing of whole portfolios (Grindley and Teece 1997, Hall and Ziedonis 2001). These industries have rapidly increased their rates of patenting in general (Hall 2003).

   Our results are consistent with the explanation that these industries acquired large numbers of software patents because they became a cost-effective means of building strategic portfolios.

2. The growth in patenting overall exhibits a sharp break in 1984 (Hall 2003). Patent propensity in the semiconductor industry also began a sharp and steady rise beginning in 1984 (Hall and Ziedonis 2001). We find a sharp rise in software patent propensity beginning in about 1984 (see Figure 1) that continues to increase at just about the same rate as Hall and Ziedonis found for all patents in the semiconductor industry.

   A consistent explanation is that the formation of the unified court of appeals for patents in the early 80s initiated changes in patent law that increased the cost effectiveness of patents, especially software patents. In industries prone to strategic patenting, this might have set off an "arms race" to build large portfolios, and obtaining software patents became a convenient way to accomplish this goal.

3. Among firms in the strategic patenting industries, we find evidence that software patents substituted for R&D during the 90s. This, too, is consistent with strategic patenting of various sorts (Bessen 2003). Maturing firms with diminished competitive advantage from technology might choose to harvest patent royalties from their past research in lieu of further R&D, especially if legal changes make patents more cost effective.[47] In turn, other firms, facing increased payment of

[47] For example, when Louis Gerstner Jr. became CEO of IBM in 1993, he slashed R&D by over $1 billion and also shifted the business much more toward services, so that the R&D-to-sales ratio has declined steadily. At the same time he initiated a much more aggressive patent-licensing strategy.

royalties, may choose to reduce R&D, possibly diverting resources to build "defensive" patent portfolios. In these cases, greater software patenting would be associated with decreased R&D intensity.

Strategic patenting provides a parsimonious explanation for our main results. Note that "strategic patenting" may involve several different forms of related behavior (see Cohen, Nelson, and Walsh 2000) and different causal mechanisms for the observed negative correlation between an increased focus on software patents and declining relative R&D intensity (e.g., diminished technological opportunity for mature firms or defensive patenting for small firms).

The *combination* of our results is difficult to reconcile with the hypothesis that software patents increased R&D incentives. It would require several coincidences: Rising patent propensity must result from a very large increase in the productivity of R&D (beginning about 1984) that occurs in only a handful of industries (but not the software industry) and yet without regard to the hardware/software distinction. It must also be the case that demand for R&D is price inelastic in those same industries, but not in the rest of the economy. This seems a rather unlikely combination.

Our interpretation might seem to conflict with the findings of Arora, Ceccagnoli, and Cohen (2003). They find that firms' ratings of "patent effectiveness," as reported in the Carnegie Mellon survey (Cohen, Nelson, and Walsh 2000) are positively correlated with a measure of patent propensity. They also find that patent effectiveness is often positively correlated with R&D spending. But Arora et al. are estimating a cross-sectional effect, while our results may reflect a shift in an industry *equilibrium* that cannot be observed in cross-sectional data. This is particularly true because strategic patenting may have characteristics of a "Prisoner's Dilemma." Policy changes may encourage firms to acquire more patents, yet, because other firms are also acquiring larger portfolios, the equilibrium result may be lower appropriability (Bessen 2003). In other words, Arora et al. show how R&D varies with patent effectiveness while we explore how changing patent policy might have changed the relationship between patent effectiveness and R&D.

VI. Conclusion

The rapid growth in software patenting has not been driven by the software publishing industry, but instead by a group of industries in computers, electronics and instruments. Other researchers have established the role of strategic patenting in these industries and strategic patenting provides a parsimonious explanation for our results. According to this theory, software patents are significant because they provide a cost-effective way for firms to build strategic patent portfolios.

Legal scholars sometimes argue that patent law should treat computer programs no differently than any other invention. This paper does not address arguments about legal consistency, but instead explores the economic effects of granting software patents in the U.S. during the 1990s. Our results are difficult to reconcile with the traditional incentive theory—that granting more patents will increase R&D investments. Rather, if legal changes have encouraged strategic patenting, the result might well be less innovation.

APPENDIX

A. Search Algorithm

The search query used is:

((“software” in specification) OR (“computer” AND “program” in specification))

AND (utility patent excluding reissues)

ANDNOT (“chip” OR “semiconductor” OR “bus” OR “circuit” OR “circuitry” in title) ANDNOT (“antigen” OR “antigenic” OR “chromatography” in specification)

Table A1. Summary Statistics

	Table 5 Sample		Total R&D Performing Sample		Table 7 Sample		Corresponding R&D Performing Sample	
	Mean	Median	Mean	Median	Mean	Median	Mean	Median
Annual successful patent applications	25.9	1			70.2	10		
Software patent applications	3.6	0			13.4%	2.7%		
Firm sales (mill. $96)	2,632.7	375.1	1,032.0	41.6	4,967.5	1,100.1	1,579.0	83.0
Firm R&D (mill. $96)	100.2	11.7	36.8	2.8	190.9	29.8	59.6	4.0
Percent not reporting R&D	27.9%		--		--		--	
Firm employees (1000s)	13.3	2.4			21.4	6.0	7.0	0.6
Percent new firms	10.0%		37.2%		10.7%		31.6%	
No. of observations	13,136		28,581		2,991		11,107	
No. of firms	1,443		4,559		489		2,351	
	1987 - 97				1991 - 97			

Note: The total R&D sample corresponding to Table 5 consists of all Compustat observations for US firms with non-missing R&D. The sample corresponding to Table 7 consists of all Compustat observations for US firms with non-missing or non-zero R&D for the current year and the 5-year lag. Sales and R&D are deflated using the GDP deflator. New firms are firms during the first five years in which they appeared in Compustat.

References

Abel, Jason R., Ernst R. Berndt, and Alan G. White, 2003 "Price Indexes for Microsoft's Personal Computer Software Products," NBER Working Paper No. 9966.

Allison, John R. and Mark A. Lemley. 2000. "Who's Patenting What? An Empirical Exploration of Patent Prosecution," *Vanderbilt Law Review*, Vol. 58, pp. 2099-2148.

Allison, John R., Mark A. Lemley, Kimberly A. Moore, and R. Derek Trunkey, 2003. "Valuable Patents," George Mason Law and Economics Research Paper No. 03-31.

Allison, John R. and Emerson H. Tiller, 2003, "Internet Business Method Patents," in Wesley M. Cohen and Stephen A. Merrill, eds., *Patents in the Knowledge-Based Economy*, National Research Council, Washington: National Academies Press, pp. 259-84.

Arora, Ashish, Marco Ceccagnoli, and Wesley M. Cohen. 2003. "R&D and the Patent Premium," NBER Working Paper No. 9431.

Berger, Philip. 1993. "Explicit and Implicit Effects of the R&D Tax Credit," *Journal of Accounting Research*, Vol. 31, pp. 131-71.

Berndt, Ernst R. 1991. *The Practice of Econometrics: Classic and Contemporary*. New York: Addison-Wesley Publishing.

Bessen, James. 2003. "Patent Thickets: Strategic Patenting of Complex Technologies," Research on Innovation Working Paper.

Bound, John, Clint Cummins, Zvi Griliches, Bronwyn H. Hall, and Adam Jaffe. 1984. "Who Does R&D and Who Patents?" in Griliches, Zvi, ed., *R&D, Patents, and Productivity*, Chicago: University of Chicago Press.

Burk, Dan L. 2002. "Patent Disclosure Doctrines: Enablement and Written Description," Public Hearings on Competition and Intellectual Property Law and Policy in the Knowledge-Based Economy. Washington: Federal Trade Commission.

Burk, Dan L., and Mark A. Lemley. 2002. "Is Patent Law Technology Specific?" *Berkeley Technology Law Journal*, Vol. 17, pp. 1155-1206.

Cohen, Julie, and Mark A. Lemley. 2001. "Patent Scope and Innovation in the Software Industry," *California Law Review*, Vol. 89, pp. 1-57.

Cohen, Wesley M., and Steven Klepper. 1996. "A Reprise of Size and R&D," *Economic Journal*, Vol. 106, pp. 925-51.

Cohen, Wesley M., Richard R. Nelson, and John P. Walsh. 2000. "Protecting Their Intellectual Assets: Appropriability Conditions and Why U.S. Manufacturing Firms Patent (or Not)," NBER Working Paper No. 7552.

Coolley, Ronald B. 1994. "The Status of Obviousness and How to Assert it as a Defense," *Journal of the Patent and Trademarks Office Society*, Vol. 76, pp. 625-44.

Cunningham, M.A. 1995. "Preliminary Injunctive Relief in Patent Litigation," *IDEA*, Vol. 35, pp. 213-59.

Dam, Kenneth W. 1995. "Some Economic Considerations in the Intellectual Property Protection of Software," *Journal of Legal Studies*, Vol. 24, pp. 321-77.

References

Dunner, Donald R., J. Michael Jakes, and Jeffrey D. Karceski. 1995. "A Statistical Look at the Federal Circuit's Patent Decisions; 1982-1994," *Federal Circuit Bar Journal*, Vol. 5, pp. 151-80.

Flynn, John D. 2001. "Comments on the International Effort to Harmonize the Substantive Requirements of Patent Laws" IBM submission at http://www.uspto.gov/web/offices/dcom/olia/harmonization/TAB42.pdf

Gallini, Nancy T. 2002. "The Economics of Patents: Lessons from Recent U.S. Patent Reform," *Journal of Economic Perspectives*, Vol. 16, pp. 131-54.

Graham, Stuart J. H., and David C. Mowery, 2003. "Intellectual Property Protection in the U. S. Software Industry," in Wesley M. Cohen and Stephen A. Merrill, eds., *Patents in the Knowledge-Based Economy*, National Research Council, Washington: National Academies Press, pp. 219-58.

Greene, William H. 1997. *Econometric Analysis*, 3rd edition. Upper Saddle River, N.J.: Prentice Hall.

Griliches, Zvi. 1990. "Patent Statistics as Economic Indicators: A Survey," *Journal of Economic Literature*, Vol. 28, pp. 1661-1707.

Griliches, Zvi, Bronwyn H. Hall, and Jerry A. Hausman. 1986. "Patents and R&D: Is There a Lag?" *International Economic Review*, Vol. 27, pp. 265-83.

Grimm, Bruce and Robert Parker. 2000. "Recognition of Business and Government Expenditures for Software as Investment: Methodology and Quantitative Impacts, 1959-98," Bureau of Economic Analysis, mimeo.

Grindley, Peter C., and David J. Teece. 1997. "Managing Intellectual Capital: Licensing and Cross-Licensing in Semiconductors and Electronics," *California Management Review*, Vol. 39, pp. 8-41.

Hahn, Robert W., and Scott Wallsten. 2003. "A Review of Bessen and Hunt's Analysis of Software Patents," AEI-Brookings Joint Center, mimeo.

Hall, Bronwyn H. 2003. "Exploring the Patent Explosion," invited lecture at the ZEW Workshop on Empirical Economics of Innovation and Patenting, Mannheim, Germany.

Hall, Bronwyn H., and John van Reenen. 2000. "How Effective Are Fiscal Incentives for R&D? A Review of the Evidence," *Research Policy*, v. 29, pp. 449-69.

Hall, Bronwyn H., Adam B. Jaffe, and Manuel Trajtenberg. 2001a. "The NBER Patent Citations Data File: Lessons, Insights and Methodological Tools," NBER Working Paper No. 8498.

Hall, Bronwyn H., Adam B. Jaffe, and Manuel Trajtenberg. 2001b. "Market Value and Patent Citations: A First Look," University of California at Berkeley Economics Department Working Paper No. E01-304.

Hall, Bronwyn H., and Rosemary Ham Ziedonis. 2001. "The Patent Paradox Revisited: An Empirical Study of Patenting in the U.S. Semiconductor Industry, 1979-1995" *RAND Journal of Economics*. Vol. 32, pp. 101-128.

References

Hausman, Jerry A., Bronwyn H. Hall, and Zvi Griliches. 1984. "Econometric Models for Count Data with an Application to the Patents-R&D Relationship," *Econometrica*, Vol. 52, pp. 909-38.

Hunt, Robert M. 1999. "Patent Reform: A Mixed Blessing for the U.S. Economy?" Federal Reserve Bank of Philadelphia *Business Review* (November/December), pp. 15-29.

Hunt, Robert M. 2001. "You Can Patent That? Are Patents on Computer Programs and Business Methods Good for the New Economy?" Federal Reserve Bank of Philadelphia *Business Review*, 1st Quarter, pp. 5-15.

Jaffe, Adam B. 2000. "The U.S. Patent System in Transition: Policy Innovation and the Innovation Process," *Research Policy*, Vol. 29, pp. 531-557.

Kemerer, Chris F. 1992 "How the Learning Curve Affects CASE Tool Adoption," IEEE Software, v. 9, pp. 23-8.

Kortum, Samuel, and Josh Lerner. 1999. "What is Behind the Recent Surge in Patenting?" *Research Policy*, Vol. 28, pp.1-22.

Lanjouw, Jean O., and Josh Lerner. 2001. "Tilting the Table? The Use of Preliminary Injunctions," *Journal of Law and Economics*, Vol. 44, pp. 573-603.

Lanjouw, Jean O., and Mark Schankerman, 2001. "Characteristics of Patent Litigation: A Window on Competition," *RAND Journal of Economics* v32, n1 (Spring 2001): 129-51.

Lerner, Josh. 1995. "Patenting in the Shadow of Competitors," *Journal of Law and Economics*, Vol. 38, pp. 463-95.

Lerner, Josh. 2004. "The New New Financial Thing: The Sources of Innovation Before and After *State Street,*" NBER Working Paper No. 10223.

Lessig, Lawrence. 2001. *The Future of Ideas: The Fate of the Commons in a Connected World*. New York: Random House.

Levin, Richard C., Alvin K. Klevorick, Richard R. Nelson, and Sidney G. Winter. 1987. "Appropriating the Returns from Industrial Research and Development," *Brookings Papers on Economic Activity*, Vol. 3, pp. 783-820.

Merges, Robert P. 1997. *Patent Law and Policy, 2nd Ed*. Charlottesville, VA: Michie Law Publishers.

Merges, Robert P. 1999. "As Many as Six Impossible Patents Before Breakfast: Property Rights for Business Concepts and Patent System Reform," *Berkeley Technology Law Journal*, Vol. 14, pp. 577-615.

National Science Foundation. 2003. *Research and Development in Industry: 2000*. Arlington, VA: National Science Foundation, Division of Science Resources Statistics.

Oliner, Stephen D., and Daniel E. Sichel. 1994, "Computers and Output Growth Revisited: How Big is the Puzzle?" *Brookings Papers on Economic Activity*, 1994:2, pp. 273-334.

Pakes, Ariel, and Zvi Griliches. 1984. "Patents and R&D at the Firm Level: A First Look," *Economic Letters*, Vol. 5, pp. 377-81.

References

Rai, Arti K. 2003. "Engaging Facts and Policy: A Multi-Institutional Approach to Patent System Reform," *Columbia Law Review*, Vol. 106, pp. 1035-1135.

Samuelson, Pamela, Randall Davis, Mitchell D. Kapor, and J.H. Reichman. 1994. "A Manifesto Concerning the Legal Protection of Computer Programs," *Columbia Law Review*, Vol. 94, pp. 2307-2431.

Scherer, Frederic M. 1965. "Firm Size, Market Structure, Opportunity, and the Output of Patented Inventions," *American Economic Review*, Vol. 55, pp. 1097-1125.

Scherer, Frederic M. 1982a. "The Office of Technology Assessment and Forecast Industry Concordance as a Means of Identifying Industry Technology Origins," *World Patent Information*, pp. 12-17.

Scherer, Frederic M., 1982b. "Demand-Pull and Technological Invention: Schmookler Revisited," *Journal of Industrial Economics*, Vol. 30, pp. 226-37.

Scherer, Frederic M. 1984. "Using Linked Patent and R&D Data to Measure Interindustry Technology Flows," in Griliches, ed., *R&D Patents and* Productivity, University of Chicago Press, pp. 417-61.

Schmookler, Jacob, 1966. *Invention and Economic Growth*, Cambridge, MA: Harvard University Press.

Silverman, Brian S. 1999. "Technological Resources and the Direction of Corporate Diversification," *Management Science*, Vol. 45, pp. 1109-1124.

Soete, Luc, 1983, "Comments on the OTAF Concordance between the US SIC and the US Patent Classification," mimeo.

USPTO. 1994. *Hearings on Software Patent Protection*, Washington: United States Patent and Trademark Office, January-February 1994.

USPTO. 1996. "Examination Guidelines for Computer Related Inventions," *Federal Register*, Vol. 61, pp. 7478-92.

Table 1. Number of Software Patents

	Patents Issued				Successful Patent Applications	
	Software Patents	Aharonian Estimates	Total Utility Patents	Software/ Total	Software Patents	Total Utility Patents
1976	765	100	70,226	1.1%	853	65,804
1977	884	100	65,269	1.4%	1,094	65,978
1978	897	150	66,102	1.4%	1,170	65,601
1979	795	200	48,854	1.6%	1,439	65,726
1980	1,080	250	61,819	1.7%	1,633	66,491
1981	1,275	300	65,771	1.9%	1,821	63,910
1982	1,402	300	57,888	2.4%	2,233	65,009
1983	1,443	350	56,860	2.5%	2,297	61,563
1984	1,939	400	67,200	2.9%	2,641	67,071
1985	2,453	500	71,661	3.4%	2,924	71,442
1986	2,657	600	70,860	3.7%	3,482	75,088
1987	3,530	800	82,952	4.3%	4,055	81,458
1988	3,495	800	77,924	4.5%	4,841	90,134
1989	4,974	1,600	95,537	5.2%	5,755	96,077
1990	4,704	1,300	90,364	5.2%	6,471	99,254
1991	5,347	1,500	96,513	5.5%	7,091	100,016
1992	5,862	1,624	97,444	6.0%	8,149	103,307
1993	6,756	2,400	98,342	6.9%	9,459	106,848
1994	8,031	4,569	101,676	7.9%	12,251	120,380
1995	9,000	6,142	101,419	8.9%	16,617	137,661
1996	11,359	9,000	109,645	10.4%	17,085	131,450
1997	12,262	13,000	111,983	10.9%	13,087	114,881
1998	19,355	17,500	147,519	13.1%		
1999	20,385	21,000	153,486	13.3%		
2000	21,065	--	157,595	13.4%		
2001	23,406	--	166,158	14.1%		
2002	24,891	--	167,438	14.9%		

Note: Utility patents excluding re-issues. Successful patent applications are the number of patent applications that resulted in patent grants by the end of 1999.

Table 2. Characteristics of Software Patents (1990-95)

	Software Patents	Other Patents
Assignee type		
Non-gov't. org. (firm)	88%	80%
Individual/ unassigned	11%	18%
Government	2%	2%
U.S. assignee (if assigned)	70%	51%
U.S. inventor	69%	53%
Mean citations received	9.7	4.6
Number of claims	16.8	12.6
Percent of self-citations	12%	13%
Percent of patents owned by top 5 percent of assignees	63%	63%

Note: Total patents: 39,700 software, 546,058 other. Self-citations is average of upper and lower bounds (see Hall, Jaffe, and Trajtenberg, 2001a). Differences between the means in the first two columns are all significant at the 1 percent level.

Table 3. Firm Characteristics by Patent Type (1990-99)

	Software Patents	Non-software Patents
Median firm market value (million $96)	24,485	11,554
Median firm sales (million $96)	13,382	8,940
Median firm R&D (million $96)	956	376
Newly public firm	5.8%	5.1%

Note: Table shows patent characteristics by type for years 1990-99 from the sample matched to Compustat. Covers 48,072 software patents and 274,529 non-software patents. Newly public firms first appeared in the Compustat file within the last 5 years.

Table 4. Successful Software Patent Applications by Industry (1994-97)

	(1)	(2)	(3)	(4)	(5)	(6)
	Share of all				All Patents/ R&D	Software patent propensity (Table 5.2)
	Software patents	Prog.	Prog. + Engineers	All patents		
Manufacturing	75%	11%	32%	88%	3.8	
Chemicals (SIC 28)	5%	1%	2%	15%	2.5	1.5
Machinery (SIC 35)	24%	3%	7%	17%	4.2	4.4
Electronics (SIC 36)	28%	2%	7%	27%	6.8	9.6
Instruments (SIC 38)	9%	1%	4%	11%	7.1	8.7
Other manu.	9%	5%	13%	18%	2.3	1.9
Non-manufacturing	25%	89%	68%	12%	3.0	
Software publishers (SIC 7372)	5%	33%	18%	1%	1.0	1.0 (all SIC 73)
Other software (SIC 737 exc. 7372, IBM)	2%			1%	2.8	
Other non-manufacturing	4%	55%	49%	4%	3.4	3.8
Addendum: IBM	6%	--	--	2%	5.0	

Note: covers 28,268 software patents and 148,552 total patents for firms in the sample of software patents matched by CUSIP to firms in Compustat. The numbers are for patent applications that were resulted in an issued patent by 1999. Programmer employment is the share of total employment accounted for by computer programmers and systems analysts in 1997 from the Occupational Employment Survey of the BLS (which includes government employees). The third column adds employment of engineers. For SIC 737 the combined employment shares also include IBM. The fifth column shows patents granted per $10 million of R&D in 1996 dollars. The last column is the exponential of the coefficient for the industry given in Table 5, column 3, normalized so that the patent propensity of SIC 73 equals one. The last row shows IBM's patents as a portion of all patents (not just those in our matched sample).

Table 5. The Propensity to Patent Software

Dependent Variable: Annual Number of Successful Software Patents Applications

	1	2	3	4	5	Annual Contribution
Ln employment	.88 (.03)*	.88 (.03)*	.88 (.03)*	.88 (.03)*	.62 (.02)*	0.5%
Ln R&D / Emp.	1.01 (.05)*	.77 (.06)*	.77 (.06)*	.78 (.05)*	.22 (.02)*	1.1%
No R&D dummy	-1.00 (.17)*	-.47 (.18)*	-.49 (.18)*	-.16 (.19)	.35 (.12)*	-0.1%
Ln Capital / Emp.	-.08 (.04)	.37 (.05)*	.37 (.05)*	.26 (.05)*	.27 (.03)*	1.8%
Programmers / Emp.		12.44 (3.13)*	12.75 (3.09)*	7.21 (.84)*	13.32 (.75)*	1.2%
Engineers / Emp.		4.13 (.55)*	4.05 (.55)*	6.57 (.44)*	6.06 (.58)*	1.2%
New firm dummy	.19 (.13)	.03 (.13)	.12 (.15)	.11 (.13)		
New firm x SIC 73			-.90 (.33)*			
Other patents / Emp.				.16 (.02)*		
Industry dummies						
SIC 28		-5.58 (.32)*	-5.52 (.31)*			
SIC 35		-4.50 (.25)*	-4.45 (.25)*			
SIC 36		-3.73 (.24)*	-3.68 (.23)*			
SIC 38		-3.83 (.25)*	-377 (.25)*			
Other manu.		-5.33 (.25)*	-5.26 (.25)*			
SIC 73		-5.99 (.63)*	-5.96 (.62)*			
Other non-manu.		-4.65 (.28)*	-4.58 (.28)*			
Firm fixed effects					Yes	
Period estimated	80 - 97	87 – 97	87 – 97	87 – 97	87 – 97	
Annual growth rate of year dummies	10.3%	9.4%	9.5%	7.2%	10.8%	
No. observations	23,108	13,136	13,136	13,136	6,587	
Log Likelihood	-68,548	-42,963	-42,870	-48,528	-9,659	

Note: Heteroscedastic-consistent standard errors in parentheses. All regressions include year dummies; columns 1 and 4 include an intercept (not shown). Asterisk indicates significance at the 1 percent level. Regressions are Poisson regressions. Column 5 has firm fixed effects (Hausman, Hall, and Griliches, 1984). R&D is deflated by the GDP deflator, capital is property, plant and equipment deflated by the NIPA capital goods deflator, and employment is in thousands. The "new firm" dummy is equal to one for the first five years a firm appears in Compustat. Other patents / employees is the ratio of patents to employees for other firms in the same 2-digit SIC industry for the given year. Annual growth rate of year dummies is year dummy for 1996 minus the year dummy for the earliest year divided by the number of years elapsed (no dummy is estimated for 1997). Annual contribution is the estimated coefficient times the difference in sample means for the variable between 1996 and 1987 divided by 9 (the number of years).

Table 6. R&D Cost Share Estimation

	$\Delta \frac{R\&D}{Cost}$	$\Delta \frac{R\&D}{Cost}$	$\Delta \ln Sales$	$\Delta \frac{R\&D}{Sales}$	$\Delta \frac{R\&D}{Sales}$	$\Delta \frac{R\&D}{Sales}$
	WLS	Heckman	WLS	WLS	Heckman	WLS
	1991-97	1991-97	1991-97	1991-97	1991-97	1985 - 97
	1	2	3	4	5	6
Δs	-.039 (.017)	-.030 (.003)*	.007 (.071)	-.037 (.006)*	-.037 (.002)*	
$\Delta \ln C$			1.018 (.024)*			
Δs x (1985-89)						.018 (.012)
Δs x (1990-93)						-.031 (.012)*
Δs x (1994-97)						-.050 (.015)*
No. total observations (firm years)	774	29,832	774	2,991	29,832	5,285
No. selected		757			2,913	
LR test of probit		$\chi^2(9) = 1059.9$ P=.000			$\chi^2(9) = 3370.6$ P=.000	
Wald test of independent equations		$\chi^2(1) = 571.6$ P=.000			$\chi^2(1) = .04$ P=.839	
Adjusted R^2	.128	--	.974	.038	--	.066

Note: Observations are firm-years, but differences are taken over five years. Robust standard errors are in parentheses. All regressions include year dummies and changes in log factor prices. The prices are for capital, labor, energy, materials, services and IT for the two-digit industry from the BLS. Asterisk indicates significance at the 1 percent level. For columns 1, 3, 4, and 6, the sample includes firms matched to patent file with positive patents and non-missing data. All regressions are weighted by $1/(1/n_{it} + 1/n_{i,t-5})$ where n is the number of patents granted. s is software share of patents applications. The Heckman sample selection equations consist of a regression equation and a selection probit. The regression coefficients are shown, the probit regressors are log market value, log sales, new firm flag and time dummies. To measure the goodness of fit for the probit, we show a likelihood ratio test that all coefficients except the constant term are zero. To determine whether sample selection biases the coefficients, we show a Wald test that the selection and regression equations are independent. In columns 4-6, observations are excluded where R&D > ½ sales (see text).

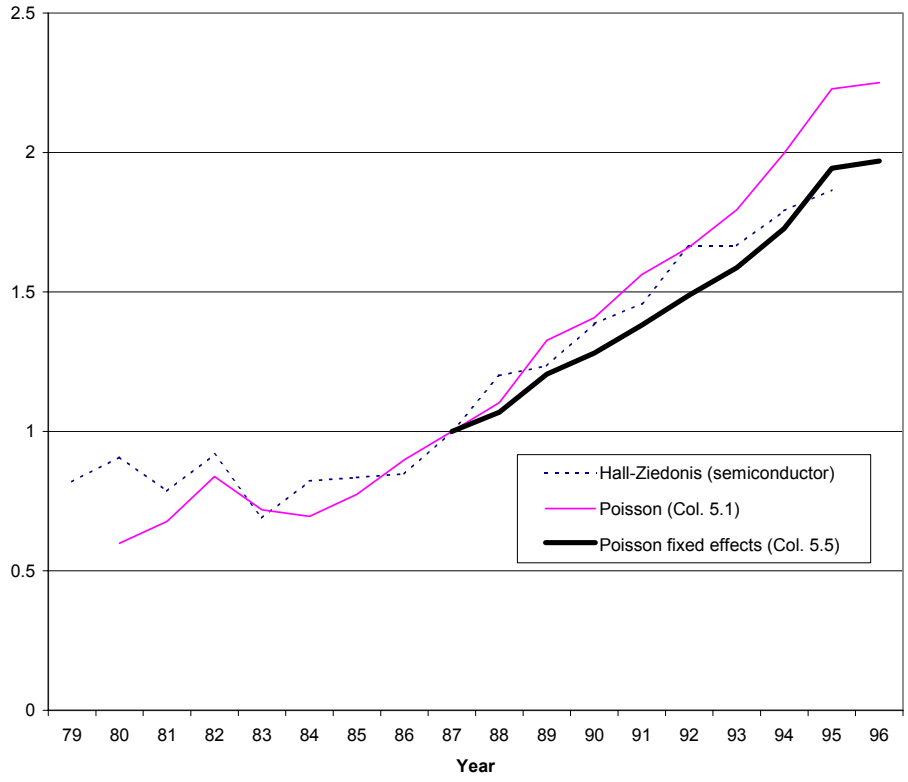
Table 7. R&D Intensity Estimation, 1991-97

Dependent Variable: $\Delta \frac{R \& D}{Sales}$

	1	2	3	4	5
Δs		.001 (.010)	-034 (.011)*	-.051 (.018)*	-.037 (.011)*
ΔProgrammers/Emp.			-.506 (.136)*		
ΔEngineers / Emp.			.120 (.087)		
Δs x (Emp. ≤ 1,000)				.007 (.030)	
Δs x (1,000<Emp. ≤ 10,000)				.039 (.022)	
Δs x (Entrant)					.002 (.028)
Δs x strategic industry		-.060 (.019)*			
Δs x SIC 28	.029 (.031)				
Δs x SIC 35	-.029 (.010)*				
Δs x SIC 36	-.090 (.026)*				
Δs x SIC 38	.027 (.028)				
Δs x Other manu.	.014 (.013)				
Δs x SIC 73	-.077 (.031)*				
Δs x other non-manu.	-.053 (.044)				
No. observations	2,991	2,991	2,967	2,991	2,991
Adjusted R^2	.060	.051	.058	.045	.043

Note: Robust standard errors in parentheses. All regressions include year dummies and changes in annual log factor prices at the two-digit industry level. Asterisk indicates significance at the 1 percent level. Time differences are five years and observations are firm years. Sample includes firms matched to patent file with positive patents and excludes observations where R&D > ½ sales. Δprogrammers/total employment and Δengineers/total employment are the changes in the shares of the respective occupational employment for the three-digit industry. All regressions are weighted by $1/\left(1/n_{it} + 1/n_{i,t-5}\right)$ where n is the number of patents granted. s is software share of patents granted, sales are deflated, and "Entrant" is a dummy for firms that entered the Compustat database after 1985. "Strategic industries" are SIC 357 (computers), SIC 36 (electrical), SIC 38 (instruments) and IBM.

Figure 1. Year Dummies, Software Patent Propensity Regressions



Source: Hall and Ziedonis (2001) and Table 5. Year dummies from Poisson regressions correspond to log relative patent propensity. Normalized to 1 in 1987.